deluge Documentation

Release 2.0b2.dev52

Deluge Team

May 21, 2018

Contents

1 The Deluge Core **3**
- 1.1 Deluge RPC . . . 3

2 Deluge's Interfaces **11**
- 2.1 Deluge GTK UI . . . 11
- 2.2 Deluge Web UI . . . 11
- 2.3 Deluge Console UI . . . 11

3 Indices and tables **13**

4 Modules **15**

Contents:

CHAPTER 1

The Deluge Core

1.1 Deluge RPC

1.1.1 Message Formats

DelugeRPC is a protocol used for daemon/client communication. There are four types of messages involved in the protocol: RPC Request, RPC Response, RPC Error and Event. All messages are zlib compressed rencoded strings and their data formats are detailed below.

RPC Request

This message is created and sent by the client to the server requesting that a remote method be called. Multiple requests can be bundled in a list.

[[request_id, method, [args], {kwargs}], . . .]

request_id (int) An integer determined by the client that is used in replies from the server. This is used to ensure the client knows which request the data is in response to. Another alternative would be to respond in the same order the requests come in, but this could cause lag if an earlier request takes longer to process.

method (str) The name of the remote method to call. This name can be in dotted format to call other objects or plugins methods.

args (list) The arguments to call the method with.

kwargs (dict) The keyword arguments to call the method with.

RPC Response

This message is created and sent in response to a RPC Request from a client. It will hold the return value of the requested method call. In the case of an error, a RPC Error message will be sent instead.

[message_type, request_id, [return_value]]

message_type (int) This will be a RPC_RESPONSE type id. This is used on the client side to determine what kind of message is being received from the daemon.

request_id (int) The request_id is the same as the one sent by the client in the initial request. It used on the client side to determine what message this is in response to.

return_value (list) The return value of the method call.

RPC Error

This message is created in response to an error generated while processing a RPC Request and will serve as a replacement for a RPC Response message.

[message_type, request_id, exception_type, exception_msg, traceback]

message_type (int) This will be a RPC_ERROR type id.

request_id (int) The request_id is the same as the one sent by the client in the initial request.

exception_type (str) The type of exception raised.

exception_msg (str) The message as to why the exception was raised.

traceback (str) The traceback of the generated exception.

Event

This message is created by the daemon and sent to the clients without being in response to a RPC Request. Events are generally sent for changes in the daemon's state that the clients need to be made aware of.

[message_type, event_name, data]

message_type (int) This will be a RPC_EVENT type id.

event_name (str) This is the name of the event being emitted by the daemon.

data (list) Additional data to be sent with the event. This is dependent upon the event being emitted.

1.1.2 Remote API

class deluge.__rpcapi.**RpcApi**

class core

Methods available in core

add_torrent_file(*filename*, *filedump*, *options*)

RPC Exported Function (*Auth Level: 5*)

Adds a torrent file to the session.

Args: filename (str): The filename of the torrent. filedump (str): A base64 encoded string of the torrent file contents. options (dict): The options to apply to the torrent upon adding.

Returns: str: The torrent_id or None.

add_torrent_file_async(*filename*, *filedump*, *options*, *save_state=True*)

RPC Exported Function (*Auth Level: 5*)

Adds a torrent file to the session asynchonously.

Args: filename (str): The filename of the torrent. filedump (str): A base64 encoded string of torrent file contents. options (dict): The options to apply to the torrent upon adding. save_state (bool): If the state should be saved after adding the file.
Returns: Deferred: The torrent ID or None.

add_torrent_files(*torrent_files*)
RPC Exported Function (*Auth Level: 5*)

Adds multiple torrent files to the session asynchonously.
Args: torrent_files (list of tuples): Torrent files as tuple of (filename, filedump, options).
Returns: Deferred

add_torrent_magnet(*uri*, *options*)
RPC Exported Function (*Auth Level: 5*)

Adds a torrent from a magnet link.
Parameters
- **uri** (*string*) – the magnet link
- **options** (*dict*) – the options to apply to the torrent on add

Returns the torrent_id
Return type string

add_torrent_url(*url*, *options*, *headers=None*)
RPC Exported Function (*Auth Level: 5*)

Adds a torrent from a url. Deluge will attempt to fetch the torrent from url prior to adding it to the session.
Parameters
- **url** (*string*) – the url pointing to the torrent file
- **options** (*dict*) – the options to apply to the torrent on add
- **headers** (*dict*) – any optional headers to send

Returns a Deferred which returns the torrent_id as a str or None

connect_peer(*torrent_id*, *ip*, *port*)
RPC Exported Function (*Auth Level: 5*)

create_account(*username*, *password*, *authlevel*)
RPC Exported Function (*Auth Level: 10*)

create_torrent(*path*, *tracker*, *piece_length*, *comment*, *target*, *webseeds*, *private*, *created_by*, *trackers*, *add_to_session*)
RPC Exported Function (*Auth Level: 5*)

disable_plugin(*plugin*)
RPC Exported Function (*Auth Level: 5*)

enable_plugin(*plugin*)
RPC Exported Function (*Auth Level: 5*)

force_reannounce(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

force_recheck(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

Forces a data recheck on torrent_ids

get_auth_levels_mappings()
RPC Exported Function (*Auth Level: 0*)

get_available_plugins()
RPC Exported Function (*Auth Level: 5*)

Returns a list of plugins available in the core

get_completion_paths(*args*)
RPC Exported Function (*Auth Level: 5*)

Returns the available path completions for the input value.

get_config()
RPC Exported Function (*Auth Level: 5*)

Get all the preferences as a dictionary

get_config_value(*key*)
RPC Exported Function (*Auth Level: 5*)

Get the config value for key

get_config_values(*keys*)
RPC Exported Function (*Auth Level: 5*)

Get the config values for the entered keys

get_enabled_plugins()
RPC Exported Function (*Auth Level: 5*)

Returns a list of enabled plugins in the core

get_external_ip()
RPC Exported Function (*Auth Level: 5*)

Returns the external ip address recieved from libtorrent.

get_filter_tree(*show_zero_hits=True*, *hide_cat=None*)
RPC Exported Function (*Auth Level: 5*)

returns {field: [(value,count)] } for use in sidebar(s)

get_free_space(*path=None*)
RPC Exported Function (*Auth Level: 5*)

Returns the number of free bytes at path
Parameters **path** (*string*) – the path to check free space at, if None, use the default download location
Returns the number of free bytes at path
Return type int
Raises **InvalidPathError** – if the path is invalid

get_known_accounts()
RPC Exported Function (*Auth Level: 10*)

get_libtorrent_version()
RPC Exported Function (*Auth Level: 5*)

Returns the libtorrent version.
Returns the version
Return type string

get_listen_port()
RPC Exported Function (*Auth Level: 5*)

Returns the active listen port

get_path_size(*path*)
RPC Exported Function (*Auth Level: 5*)

Returns the size of the file or folder 'path' and -1 if the path is unaccessible (non-existent or insufficient privs)

get_proxy()
RPC Exported Function (*Auth Level: 5*)

Returns the proxy settings
Returns: dict: Contains proxy settings.
Notes:
Proxy type names: 0: None, 1: Socks4, 2: Socks5, 3: Socks5 w Auth, 4: HTTP, 5: HTTP w Auth, 6: I2P

get_session_state()
RPC Exported Function (*Auth Level: 5*)

Returns a list of torrent_ids in the session.

get_session_status(*keys*)
RPC Exported Function (*Auth Level: 5*)
Gets the session status values for 'keys', these keys are taking from libtorrent's session status.

See: http://www.rasterbar.com/products/libtorrent/manual.html#status
param keys the keys for which we want values
type keys list
returns a dictionary of {key: value, ...}
rtype dict

get_torrent_status(*torrent_id*, *keys*, *diff=False*)
RPC Exported Function (*Auth Level: 5*)

get_torrents_status(*filter_dict*, *keys*, *diff=False*)
RPC Exported Function (*Auth Level: 5*)

returns all torrents , optionally filtered by filter_dict.

glob(*path*)
RPC Exported Function (*Auth Level: 5*)

move_storage(*torrent_ids*, *dest*)
RPC Exported Function (*Auth Level: 5*)

pause_session()
RPC Exported Function (*Auth Level: 5*)

Pause all torrents in the session

pause_torrent(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

queue_bottom(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

queue_down(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

queue_top(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

queue_up(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

remove_account(*username*)
RPC Exported Function (*Auth Level: 10*)

remove_torrent(*torrent_id*, *remove_data*)
RPC Exported Function (*Auth Level: 5*)

Removes a single torrent from the session.

Args: torrent_id (str): The torrent ID to remove. remove_data (bool): If True, also remove the downloaded data.

Returns: bool: True if removed successfully.

Raises: InvalidTorrentError: If the torrent ID does not exist in the session.

remove_torrents(*torrent_ids*, *remove_data*)
RPC Exported Function (*Auth Level: 5*)

Remove multiple torrents from the session.

Args: torrent_ids (list): The torrent IDs to remove. remove_data (bool): If True, also remove the downloaded data.

Returns:

list: An empty list if no errors occurred otherwise the list contains tuples of strings, a torrent ID and an error message. For example:

[('<torrent_id>', 'Error removing torrent')]

rename_files(*torrent_id*, *filenames*)
RPC Exported Function (*Auth Level: 5*)

Rename files in torrent_id. Since this is an asynchronous operation by libtorrent, watch for the TorrentFileRenamedEvent to know when the files have been renamed.

Parameters

- **torrent_id** (*string*) – the torrent_id to rename files
- **filenames** (*((index, filename), ..)*) – a list of index, filename pairs

Raises **InvalidTorrentError** – if torrent_id is invalid

rename_folder(*torrent_id*, *folder*, *new_folder*)
RPC Exported Function (*Auth Level: 5*)

Renames the 'folder' to 'new_folder' in 'torrent_id'. Watch for the TorrentFolderRenamedEvent which is emitted when the folder has been renamed successfully.

Parameters

- **torrent_id** (*string*) – the torrent to rename folder in
- **folder** (*string*) – the folder to rename
- **new_folder** (*string*) – the new folder name

Raises **InvalidTorrentError** – if the torrent_id is invalid

rescan_plugins()
RPC Exported Function (*Auth Level: 5*)

Rescans the plugin folders for new plugins

resume_session()
RPC Exported Function (*Auth Level: 5*)

Resume all torrents in the session

resume_torrent(*torrent_ids*)
RPC Exported Function (*Auth Level: 5*)

set_config(*config*)
RPC Exported Function (*Auth Level: 5*)

Set the config with values from dictionary

set_torrent_auto_managed(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'auto_managed'

set_torrent_file_priorities(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'file_priorities'

set_torrent_max_connections(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'max_connections'

set_torrent_max_download_speed(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'max_download_speed'

set_torrent_max_upload_slots(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'max_upload_slots'

set_torrent_max_upload_speed(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'max_upload_speed'

set_torrent_move_completed(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'move_completed'

set_torrent_move_completed_path(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'move_completed_path'

set_torrent_options(*torrent_ids*, *options*)
RPC Exported Function (*Auth Level: 5*)

Sets the torrent options for torrent_ids

Args: torrent_ids (list): A list of torrent_ids to set the options for. options (dict): A dict of torrent options to set. See torrent.TorrentOptions class for valid keys.

set_torrent_prioritize_first_last(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'prioritize_first_last'

set_torrent_remove_at_ratio(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'remove_at_ratio'

set_torrent_stop_at_ratio(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'stop_at_ratio'

set_torrent_stop_ratio(***kwargs*)
RPC Exported Function (*Auth Level: 5*)

Deprecated: Use set_torrent_options with 'stop_ratio'

set_torrent_trackers (*torrent_id*, *trackers*)
RPC Exported Function (*Auth Level: 5*)

Sets a torrents tracker list. trackers will be [{"url", "tier"}]

test_listen_port ()
RPC Exported Function (*Auth Level: 5*)

Checks if the active port is open
Returns True if the port is open, False if not
Return type bool

update_account (*username*, *password*, *authlevel*)
RPC Exported Function (*Auth Level: 10*)

upload_plugin (*filename*, *filedump*)
RPC Exported Function (*Auth Level: 5*)
This method is used to upload new plugins to the daemon. It is used when connecting to the daemon remotely and installing a new plugin on the client side. 'plugin_data' is a xmlrpc.Binary object of the file data, ie, plugin_file.read()

class daemon
Methods available in daemon

authorized_call (*rpc*)
RPC Exported Function (*Auth Level: 1*)

Determines if session auth_level is authorized to call RPC.
Args: rpc (str): A RPC, e.g. core.get_torrents_status
Returns: bool: True if authorized to call RPC, otherwise False.

get_method_list ()
RPC Exported Function (*Auth Level: 5*)

Returns a list of the exported methods.

shutdown (**args*, ***kwargs*)
RPC Exported Function (*Auth Level: 5*)

CHAPTER 2

Deluge's Interfaces

Interfaces.

2.1 Deluge GTK UI

2.2 Deluge Web UI

The Deluge web interface is intended to be a full featured interface built using the ExtJS framework, running on top of a Twisted webserver.

2.2.1 SSL Configuration

By default the web interface will use the same private key and certificate as the Deluge daemon. If you wish to use a different certificate/key (see How to Create a SSL Certificate for information on creating one) you are able to specify which you want to use.

There are 2 ways to enable SSL encryption in the webserver, 1 is to specify it in your configuration (accessible via the Preferences window). The other is to add '–ssl' when running the webserver, which will override the configuration value and enable SSL.

2.3 Deluge Console UI

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

CHAPTER 4

Modules

Index

A

add_torrent_file() (deluge.__rpcapi.RpcApi.core method), 4
add_torrent_file_async() (deluge.__rpcapi.RpcApi.core method), 4
add_torrent_files() (deluge.__rpcapi.RpcApi.core method), 5
add_torrent_magnet() (deluge.__rpcapi.RpcApi.core method), 5
add_torrent_url() (deluge.__rpcapi.RpcApi.core method), 5
authorized_call() (deluge.__rpcapi.RpcApi.daemon method), 10

C

connect_peer() (deluge.__rpcapi.RpcApi.core method), 5
create_account() (deluge.__rpcapi.RpcApi.core method), 5
create_torrent() (deluge.__rpcapi.RpcApi.core method), 5

D

disable_plugin() (deluge.__rpcapi.RpcApi.core method), 5

E

enable_plugin() (deluge.__rpcapi.RpcApi.core method), 5

F

force_reannounce() (deluge.__rpcapi.RpcApi.core method), 5
force_recheck() (deluge.__rpcapi.RpcApi.core method), 5

G

get_auth_levels_mappings() (deluge.__rpcapi.RpcApi.core method), 5
get_available_plugins() (deluge.__rpcapi.RpcApi.core method), 5
get_completion_paths() (deluge.__rpcapi.RpcApi.core method), 6
get_config() (deluge.__rpcapi.RpcApi.core method), 6
get_config_value() (deluge.__rpcapi.RpcApi.core method), 6
get_config_values() (deluge.__rpcapi.RpcApi.core method), 6
get_enabled_plugins() (deluge.__rpcapi.RpcApi.core method), 6
get_external_ip() (deluge.__rpcapi.RpcApi.core method), 6
get_filter_tree() (deluge.__rpcapi.RpcApi.core method), 6
get_free_space() (deluge.__rpcapi.RpcApi.core method), 6
get_known_accounts() (deluge.__rpcapi.RpcApi.core method), 6
get_libtorrent_version() (deluge.__rpcapi.RpcApi.core method), 6
get_listen_port() (deluge.__rpcapi.RpcApi.core method), 6
get_method_list() (deluge.__rpcapi.RpcApi.daemon method), 10
get_path_size() (deluge.__rpcapi.RpcApi.core method), 6
get_proxy() (deluge.__rpcapi.RpcApi.core method), 7
get_session_state() (deluge.__rpcapi.RpcApi.core method), 7
get_session_status() (deluge.__rpcapi.RpcApi.core method), 7
get_torrent_status() (deluge.__rpcapi.RpcApi.core method), 7
get_torrents_status() (deluge.__rpcapi.RpcApi.core method), 7
glob() (deluge.__rpcapi.RpcApi.core method), 7

M

move_storage() (deluge.__rpcapi.RpcApi.core method), 7

P

pause_session() (deluge.__rpcapi.RpcApi.core method), 7
pause_torrent() (deluge.__rpcapi.RpcApi.core method), 7

Q

queue_bottom() (deluge.__rpcapi.RpcApi.core method), 7
queue_down() (deluge.__rpcapi.RpcApi.core method), 7
queue_top() (deluge.__rpcapi.RpcApi.core method), 7
queue_up() (deluge.__rpcapi.RpcApi.core method), 7

R

remove_account() (deluge.__rpcapi.RpcApi.core method), 7
remove_torrent() (deluge.__rpcapi.RpcApi.core method), 7
remove_torrents() (deluge.__rpcapi.RpcApi.core method), 8
rename_files() (deluge.__rpcapi.RpcApi.core method), 8
rename_folder() (deluge.__rpcapi.RpcApi.core method), 8
rescan_plugins() (deluge.__rpcapi.RpcApi.core method), 8
resume_session() (deluge.__rpcapi.RpcApi.core method), 8
resume_torrent() (deluge.__rpcapi.RpcApi.core method), 8
RpcApi (class in deluge.__rpcapi), 4
RpcApi.core (class in deluge.__rpcapi), 4
RpcApi.daemon (class in deluge.__rpcapi), 10

S

set_config() (deluge.__rpcapi.RpcApi.core method), 8
set_torrent_auto_managed() (deluge.__rpcapi.RpcApi.core method), 8
set_torrent_file_priorities() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_max_connections() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_max_download_speed() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_max_upload_slots() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_max_upload_speed() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_move_completed() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_move_completed_path() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_options() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_prioritize_first_last() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_remove_at_ratio() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_stop_at_ratio() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_stop_ratio() (deluge.__rpcapi.RpcApi.core method), 9
set_torrent_trackers() (deluge.__rpcapi.RpcApi.core method), 9
shutdown() (deluge.__rpcapi.RpcApi.daemon method), 10

T

test_listen_port() (deluge.__rpcapi.RpcApi.core method), 10

U

update_account() (deluge.__rpcapi.RpcApi.core method), 10
upload_plugin() (deluge.__rpcapi.RpcApi.core method), 10